Bornes inférieures à base d'inégalités valides pour les WCSP*

Mohand Ou Idir Khemmoudj, Hachemi Bennaceur

LIPN-CNRS UMP 7030 Av J-B Clément 93430 Villetaneuse France
{MohandOuIdir.Khemmoudj,Hachemi.Bennaceur}@lipn.fr

Résumé

La plus part des algorithmes de résolution efficace de WCSP se basent sur la notion de consistance d'arc utilisée pour transformer un WCSP en un WCSP équivalent et plus facile à résoudre. Dans ce but, plusieurs formes de consistance d'arc ont été proposées : AC* [9], DAC* [8], FDAC* [8], EDAC* [4]. Récemment, une consistance d'arc optimale (OSAC pour Optimal Soft Arc Consistency) [2] a été proposée. Elle se base sur la résolution d'un Programme Linéaire. Son inconvénient réside dans le fait qu'elle nécessite beaucoup de temps de calcul. Cet inconvénient est dû à la taille du programme linéaire résolu.

Nous proposons une nouvelle technique de transformation d'un WCSP en un WCSP équivalent. Cette technique se base sur la modélisation du WCSP sous forme d'un programme linéaire plus facile à résoudre que le calcul de OSAC.

Abstract

Most of efficient WCSP solving methods are based on arc consistency notion used to transform a WCSP into an equivalent one easier to solve. There are several forms of arc consistency : AC* [9], DAC* [8], FDAC* [8], EDAC* [4]. Recently, an Optimal Soft Arc Consistency (OSAC) was proposed [2]. But this technique requires much computing time since it is based on a large linear program.

We propose in this work a new technique to transform a WCSP into an equivalent one. Our technique is based on modeling of the WCSP as a linear program easier to solve than the computing of OSAC.

1 Introduction

Le problème WCSP (*Weighted Constraint Satisfaction Problem*) consiste à satisfaire de manière optimale un ensemble de contraintes pondérées.

*Ce travail est en partie soutenu par Electricté de France (EDF).

La plus part des algorithmes de résolution efficace de WCSP se basent sur la notion de consistance d'arc utilisée pour transformer un WCSP en un WCSP équivalent et plus facile à résoudre. Dans ce but, plusieurs formes de consistance d'arc ont été proposées : AC* [9], DAC* [8], FDAC* [8], EDAC* [4]. Récemment, une consistance d'arc optimale (OSAC pour Optimal Soft Arc Consistency) [2] a été proposée. Elle se base sur la résolution d'un Programme Linéaire. Son inconvénient réside dans le fait qu'elle nécessite beaucoup de temps de calcul. Cet inconvénient est dû à la taille du programme linéaire résolu.

Nous présentons dans cet article une nouvelle technique de transformation d'un WCSP binaire en un WCSP équivalent. Cette technique se base sur la notion d'inégalité valide qui exprime une relaxation d'une contrainte. Nous démontrons qu'il est toujours possible de construire un système linéaire de valeur égale à la borne OSAC et contenant une inégalité valide par contrainte du WCSP. La construction d'un tel système optimal étant coûteuse en temps de calcul, nous proposons alors une autre manière de construire un bon système linéaire qui n'est pas forcément optimal. Ce système linéaire est obtenu par la résolution de la relaxation continue d'un programme linéaire à base d'inégalités valides équivalent au WCSP.

Nous avons évalué expérimentalement cette idée, dans le cadre de prétraitement, sur des instances du problème d'allocation de fréquences aux liens radios et sur des Max-CSP aléatoires. Les résultats obtenus montrent bien l'intérêt pratique de la technique proposée. En plus de cet intérêt pratique, les résultats théoriques présentés dans cet article ouvrent de nouvelles voies de recherche pour l'exploitation de la notion d'inégalité valide dans le cadre des WCSP.

2 Préliminaires

Dans cette section, nous présentons le formalisme WCSP ainsi qu'une représentation graphique utilisée dans la suite puis nous décrivons une formulation mathématique de WCSP proposée dans la littérature.

2.1 Formalisme WCSP

Formellement, un WCSP est la donnée d'un quadruplet (X, D, C, Sv) :

1. $Sv = (\{0, 1, ..., h\}, \oplus, <)$ est une structure de valuation :
   - $<$ est la relation d'ordre usuelle sur les entiers ;
   - h est un entier (c'est une borne supérieure) utilisé pour exprimer l'interdiction d'instanciations partielles ou complètes ;
   - $\oplus$ est une loi de composition interne commutative, associative :
     - $a \oplus b = \min(a + b, h)$;
     - $a \ominus b = \begin{cases} a - b & \text{si } a \neq h; \\ h & \text{sinon;} \end{cases}$
2. $X = \{X_1, X_2, \ldots, X_n\}$ est un ensemble de n variables ;
3. $D = \{D_1, D_2, \ldots, D_n\}$ est un ensemble de n domaines où chaque D_i représente l'ensemble de $d_i \leq d$ valeurs possibles pour la variable X_i ;
4. C est un ensemble composé d'une contrainte $c_\emptyset$ d'arité nulle (c'est une constante), de n contraintes unaires[1] $c_1, c_2, ..., c_n$ et e contraintes binaires. Si X_i et X_j sont les variables sur lesquelles porte une contrainte binaire alors cette dernière est notée c_{ij}. Les contraintes unaires et binaires sont des fonctions de coût : une contrainte unaire c_i associe un coût $c_i(k) \in \{0, 1, ..., h\}$ à chaque valeur $k \in D_i$ de la variable X_i. Si $c_i(k) = h$ alors on dit que la valeur $k \in D_i$ est interdite par la contrainte c_i. Une contrainte binaire c_{ij} associe un coût $c_{ij}(k, l) \in \{0, 1, ..., h\}$ à chaque couple $(k, l) \in D_i \times D_j$. Si $c_{ij}(k, l) = h$ alors on dit que le couple (k, l) est interdit par la contrainte c_{ij}.

La résolution d'un WCSP consiste à trouver une instanciation complète $I = (v_1, v_2, ..., v_n) \in \prod_{X_i \in X} D_i$ de coût $V(I) = c_\emptyset \oplus \sum_{X_i \in X} c_i(v_i) \oplus \sum_{c_{ij} \in C} c_{ij}(v_i, v_j)$ inférieur à h et inférieur ou égal au coût $V(I')$ de toute autre instanciation complète $I' \in \prod_{X_i \in X} D_i$.

L'entier h peut être initililisé par le coût d'une instanciation complète trouvée de manière heuristique. Dans la suite, pour faciliter la présentation on se met dans le cas où h est infini. Dans ce cas, les lois $\oplus$ et $\ominus$ sont équivalentes aux lois usuelles d'addition et de soustraction.

[1] $c_\emptyset, c_1, c_2, ..., c_n$ sont éventuellement des fonctions nulles.

2.2 Représentation graphique de WCSP

Dans la suite on associera à un WCSP un graphe $G(V, A)$ où V est l'ensemble des sommets et A l'ensemble des arêtes. L'ensemble V est subdivisé en n sous-ensembles $V_1, V_2, ..., V_n$. Un ensemble V_i contient un sommet (i, k) par valeur $k \in D_i$, i.e, $V_i = \{(1, k) : k \in D_i\}$. Si (i, k) est un sommet de V_i tel que $c_i(k) > 0$ alors il sera étiqueté par $c_i(k)$.

Dans A, il existe une arête entre deux sommets (i, k) et (j, l) si $c_{ij}(k, l) \geq 1$. On notera $< (i, k), (j, l) >$ l'arête qui lie les deux sommets (i, k) et (j, l). Si $c_{ij}(k, l) \geq 2$ alors l'arête $< (i, k), (j, l) >$ sera étiquetée par $c_{ij}(k, l)$.

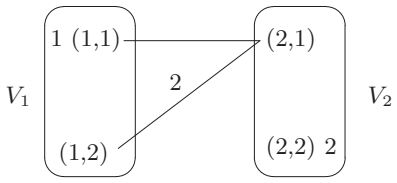

FIG. 1 – Exemple de WCSP.

Exemple 1 *La figure 1 montre un exemple de graphe $G(V, A)$ représentant un WCSP avec deux variables ($X = \{X_1, X_2\}$) qui ont le même domaine $\{1, 2\}$ et trois contraintes ($C = \{c_1, c_2, c_{12}\}$). L'ensemble V est composé des deux sous-ensembles V_1 et V_2 associés aux variables : à chaque variable X_i est associé l'ensemble $V_i = \{(i, 1), (i, 2)\}$ de sommets. Les sommets $(1, 1)$ et $(2, 2)$ sont étiquetés par $c_1(1) = 1$ et $c_2(2) = 2$ respectivement. Les étiquettes des autres sommets sont nulles et ne sont pas représentées. L'ensemble A est composé de deux arêtes $A = \{< (1, 1), (2, 1) >, < (1, 2), (2, 1) >\}$. L'étiquette 2 de l'arête $< (1, 2), (2, 1) >$ signifie que $c_{12}(2, 1) = 2$. L'arête $< (1, 1), (2, 1) >$ n'est pas étiquetée. Sa présence signifie que $c_{12}(1, 1) = 1$.*

2.3 Formulation mathématique de WCSP

Dans [7] une formulation linéaire en nombres 0-1 est proposée pour les WCSP. Nous noterons $KPLNE$ cette formulation. Elle introduit pour chaque variable $X_i \in X$, d_i variables binaires $x_i(1), x_i(2), ..., x_i(d_i)$. Ces variables doivent satisfaire les contraintes d'affectation suivantes :

$$(S) \begin{cases} \sum_{k \in D_i} x_i(k) = 1 & \forall X_i \in X \\ x_i(k) \in \{0, 1\} & \forall X_i \in X, \ \forall k \in D_i \end{cases}$$

Le système (S) ci-dessus exprime le fait que parmi les variables binaires associées à une variable à domaine fini du WCSP, exactement une d'entre elles doit prendre la valeur 1 (dans le graphe $G = (V, A)$ associé au WCSP, un seul sommet doit être sélectionné dans chaque ensemble $V_i \in V$).

Remarque 1 *Les contraintes*

$$X_i = \sum_{k \in D_i} k x_i(k) \;\; \forall X_i \in X$$

peuvent être ajoutées au système (S) pour exprimer explicitement le lien entre les variables binaires introduites et les variables à domaines finis du WCSP.

Dans $KPLNE$, on introduit aussi, pour chaque contrainte $c_{ij} \in C$ et pour chaque couple de valeurs $(k, l) \in D_i \times D_j$, la variable binaire $y_{ij}(k, l)$. Ainsi, le WCSP peut être exprimé comme suit :

$$KPLNE \begin{cases} \min c_\emptyset + f_1(x) + f_2(y) \\ x_i(k) = \sum\limits_{l \in D_j} y_{ij}(k,l) \;\; \forall c_{ij} \in C, \forall k \in D_i \\ x_j(l) = \sum\limits_{k \in D_i} y_{ij}(k,l) \;\; \forall c_{ij} \in C, \forall l \in D_j \\ x \in S \end{cases}$$

avec :

- x est le vecteur des variables binaires $x_i(k),\ \forall X_i \in X, \forall k \in D_i$;
- y est le vecteur des variables binaires $y_{ij}(k, l),\ \forall c_{ij} \in C, \forall (k, l) \in D_i \times D_j$;
- $f_1(x) = \sum\limits_{i=1}^{n} \sum\limits_{k \in D_i} c_i(k).x_i(k)$ exprime les contraintes unaires du WCSP : une contrainte unaire c_i associe le coût $c_i(k)$ à l'affectation $X_i = k$. Cette information est représentée par le produit $c_i(k)x_i(k)$;
- $f_2(y) = \sum\limits_{c_{ij} \in C} \sum\limits_{k \in D_i} \sum\limits_{l \in D_j} c_{ij}(k,l) y_{ij}(k,l)$ exprime les contraintes binaires du WCSP : une contrainte binaire c_{ij} associe le coût $c_{ij}(k, l)$ à l'affectation $(X_i = k, X_j = l)$. Cette information est représentée par le produit $c_{ij}(k, l)y_{ij}(k, l)$;
- les contraintes $x_i(k) = \sum\limits_{l \in D_j} y_{ij}(k,l)$ et $x_j(l) = \sum\limits_{k \in D_i} y_{ij}(k,l)$ contraignent la variable $y_{ij}(k, l)$ à prendre la valeur 1 si et seulement si $x_i(k) = 1$ et $x_j(l) = 1$, c'est-à-dire si $(X_i = k, X_j = l)$.

Nous noterons KPL la relaxation continue de $KPLNE$. Cette relaxation (KPL) peut être résolue en un temps polynomial [5].

Récemment, un résultat intéressant est présenté dans [2]. Dans cet article, les auteurs ont proposé un modèle linéaire à variables continues et démontrent que sa résolution permet de calculer une borne inférieure (OSAC, Optimal Soft Arc Consistency) meilleure que celles calculées par les techniques se basant sur l'arc-consistance telles que FDAC* et EDAC*. La formulation qu'ils proposent est le dual de KPL.

L'inconvénient de la formulation $KPLNE$ est qu'elle introduit beaucoup de variables et de contraintes ($O(nd + ed^2)$ variables et $O(2ed + n)$ contraintes).

3 Modèles linéaires à base d'inégalités valides

Dans [6] nous avons introduit la notion de clique binaire et nous avons montré comment l'exploiter dans le cadre de la résolution de Max-CSP. Nous rappelons qu'une clique binaire Γ_{ij} associée à une contrainte c_{ij} est une union de deux sous-ensembles E_i et E_j de V_i et V_j respectivement telle que :

$$c_{ij}(k, l) \neq 0 \vee (i, k) \notin E_i \vee (j, l) \notin E_j, \forall (k, l) \in D_i \times D_j.$$

Notons que, par abus de notation, pour chaque contrainte c_{ij}, E_i désigne un sous-ensemble de V_i qui forme une clique avec E_j suivant la contrainte c_{ij}.

Une clique binaire Γ_{ij} est maximale s'il n'existe pas une autre clique binaire Γ'_{ij} telle que $\Gamma'_{ij} \supset \Gamma_{ij}$. Dans le cadre des Max-CSP, un ensemble Γ de cliques binaires est dit complet si et seulement si pour toute contrainte c_{ij} et pour tout couple $(k, l) \in D_i \times D_j : c_{ij}(k, l) \neq 0$, l'ensemble Γ contient au moins une clique binaire $\Gamma_{ij} = E_i \cup E_j$ associée à c_{ij} telle que $(i, k) \in E_i$ et $(j, l) \in E_j$.

Nous introduisons dans cette section la notion d'inégalité binaire valide qui généralise celle de clique binaire et nous montrons comment l'exploiter pour définir des modèles linéaires utiles pour la résolution de WCSP. Dans ces modèles, une variable η_{ij} est introduite pour chaque contrainte binaire $c_{ij} \in C$. Le vecteur des variables η_{ij} sera noté η.

Définition 1 (Inégalité binaire valide) *Soit c_{ij} une contrainte d'un WCSP, on appellera inégalité binaire associée à c_{ij} toute inégalité de la forme*

$$\sum_{k \in D_i} a_{ij}(i, k) x_i(k) + \sum_{l \in D_j} a_{ij}(j, l) x_j(l) \leq b_{ij} + \eta_{ij} \quad (1)$$

où $a_{ij}(i, k) \in \mathbb{R}\ \forall k \in D_i$, $a_{ij}(j, l) \in \mathbb{R}\ \forall l \in D_j$ et $b_{ij} \in \mathbb{R}$. C'est une inégalité qui porte sur les variables binaires modélisant les valeurs possibles des deux variables X_i et X_j et sur la variable de coût $\eta_{ij} \geq 0$ associée à c_{ij}. Cette inégalité sera dite valide si et seulement si on a $a_{ij}(i, k) + a_{ij}(j, l) \leq b_{ij} + c_{ij}(k, l)$ quelque soit le couple $(k, l) \in D_i \times D_j$. Si les coefficients $b_{ij}; a_{ij}(i, k)\ \forall k \in D_i; a_{ij}(j, l)\ \forall l \in D_j$ sont tous

entiers et tels que pour tout couple $(k,l) \in D_i \times D_j$ *on a :*

$$c_{ij}(k,l) \neq 0 \vee a_{ij}(i,k) = 0 \vee a_{ij}(j,l) = 0$$

alors on dira que cette inégalité est une clique binaire pondérée.

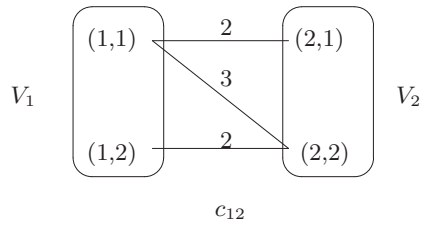

FIG. 2 – Exemple d'une contrainte pondérée.

Exemple 2 *À la contrainte représentée par la figure 2 on peut associer les deux inégalités valides* $2x_1(1) + x_1(2) + x_2(1) + 2x_2(2) \leq 2 + \eta_{12}$ *et* $3x_1(1) + \frac{3}{2}x_1(2) + \frac{3}{2}x_2(1) + 3x_2(2) \leq 3 + \eta_{12}$*. Aucune de ces deux inégalités n'est une clique binaire pondérée car les sommets* $(1,2)$ *et* $(2,1)$ *ne sont pas adjacents et les variables* $x_1(2)$ *et* $x_2(1)$ *apparaissent toutes les deux dans les deux inégalités avec des poids non nuls. Le fait que les coefficients de la dernière ne sont pas tous entiers est une autre raison qui fait qu'elle n'est pas considérée comme étant une clique binaire pondérée.*

Remarque 2 *Une inégalité valide de type (1) exprime une relaxation de la contrainte* c_{ij} *: c'est une inégalité qu'on peut toujours satisfaire avec un coût* η_{ij} *inférieur ou égal au coût relatif à* c_{ij}*.*

Pour toute contrainte du WCSP, il est possible d'associer plusieures inégalités valides. Dans la suite, on dira qu'un système $Ax \leq b + \eta$ est valide s'il est composé d'inégalités toutes valides.

Théorème 1 *Si* $Ax \leq b + \eta$ *est un système linéaire dont toutes les contraintes sont des inégalités valides de type (1) alors toute borne inférieure du programme*

$$IP(A,b) \begin{cases} \min c_\emptyset + \sum_{i=1}^{n} \sum_{k \in D_i} c_i(k)x_i(k) + \sum_{c_{ij} \in C} \eta_{ij} \\ s.c : Ax \leq b + \eta, x \in S, \eta \geq 0 \end{cases}$$

est une borne inférieure du WCSP.

Démonstration 1 *Puisque le système* $Ax \leq b+\eta$ *est valide alors à une solution optimale* $I = (v_1, v_2, ..., v_n)$ *du WCSP on peut associer la solution* (x, η) *de* $IP(A,b)$ *telle que* $x_i(v_i) = 1 \, \forall X_i \in X; x_i(k) = 0 \, \forall X_i \in X, \forall k \in D_i \setminus \{v_i\}$ *et* $\eta_{ij} = c_{ij}(v_i, v_j) \, \forall c_{ij} \in C$*. La solution* (x, η) *n'est pas forcément optimale et son coût* $V(x, \eta)$ *est par construction égal au coût* $V(I)$ *de* I*. La valeur optimale de* $IP(A,b)$ *est donc forcément inférieure ou égale à* $V(I)$*, il en est de même pour toute borne inférieure de* $IP(A,b)$*.*

Corollaire 1 *Si* $Ax \leq b+\eta$ *est valide et si pour toute contrainte* c_{ij} *du WCSP et pour tout couple* $(k,l) \in D_i \times D_j$*, le système* $Ax \leq b+\eta$ *contient au moins une inégalité associée à* c_{ij} *telle que*

$$a_{ij}(i,k) + a_{ij}(j,l) = b_{ij} + c_{ij}(k,l)$$

alors $IP(A,b)$ *est équivalent au WCSP.*

4 Utilisation d'inégalités valides pour le prétraitement de WCSP

Nous proposons dans cette section une technique exploitant des inégalités valides pour le prétraitement de WCSP.

4.1 Opérations de transformation valides

Nous montrons dans cette sous-section comment exploiter un système contenant **une seule** inégalité valide de type (1) par contrainte du WCSP pour le transformer en un autre WCSP équivalent. Le choix du système à utiliser sera aborder par la suite.

Soit $Ax \leq b + \eta$ un système contenant **une seule** inégalité valide de type (1) par contrainte $c_{ij} \in C$ du WCSP et considérons les opérations suivantes :

op1 $c_{ij}(k,l) \leftarrow c_{ij}(k,l) + b_{ij} - a_{ij}(i,k) - a_{ij}(j,l) \, \forall c_{ij} \in C, \, \forall (k,l) \in D_i \times D_j$;

op2 $c_i(k) \leftarrow c_i(k) + \sum_{j : c_{ij} \in C} a_{ij}(i,k) \, \forall X_i \in X, \, \forall k \in D_i$;

op3 $u_i \leftarrow \min_{k \in D_i} c_i(k) \, \forall X_i \in X$;

op4 $c_i(k) \leftarrow c_i(k) - u_i \, \forall X_i \in X, \, \forall k \in D_i$;

op5 $c_\emptyset \leftarrow c_\emptyset + \sum_{i=1}^{n} u_i - \sum_{c_{ij} \in C} b_{ij}$.

Ces opérations peuvent être vues comme une suite d'opérations de projection classique dirigées par le système linéaire $Ax \leq b + \eta$ d'inégalités valides : pour toute variable $X_i \in X$ et toute contrainte $c_{ij} \in C$, b_{ij} est une quantité projetée de $c_\emptyset$ sur c_{ij}, u_i est une quantité projetée de c_i sur $c_\emptyset$, $a_{ij}(i,k) \, (\forall k \in D_i)$ est une quantité projetée de c_{ij} sur $c_i(k)$ et $a_{ij}(j,l) \, (\forall l \in D_j)$ est une quantité projetée de c_{ij} sur $c_j(l)$.

Théorème 2 *Les opérations ci-dessus (op1, op2, op3, op4 et op5) sont valides : elles transforment le WCSP en un autre WCSP équivalent.*

Démonstration 2 *Il suffit de remarquer que le coût de toute instanciation complète* $I = (X_1 = v_1, X_2 = v_2, ..., X_n = v_n)$ *reste le même après transformation :*

$$
\begin{aligned}
& c_\emptyset + \sum_{i=1}^{n} u_i - \sum_{c_{ij} \in C} b_{ij} \\
& + \sum_{i=1}^{n} \left(c_i(v_i) + \sum_{j : c_{ij} \in C} a_{ij}(i, v_i) - u_i \right) \\
& + \sum_{c_{ij} \in C} (c_{ij}(v_i, v_j) + b_{ij} - a_{ij}(i, v_i) - a_{ij}(j, v_j)) \\
& = c_\emptyset + \sum_{i=1}^{n} c_i(v_i) + \sum_{c_{ij} \in C} c_{ij}(v_i, v_j) \\
& + \sum_{i=1}^{n} u_i - \sum_{i=1}^{n} u_i - \sum_{c_{ij} \in C} b_{ij} + \sum_{c_{ij} \in C} b_{ij} \\
& + \sum_{i=1}^{n} \sum_{j : c_{ij} \in C} a_{ij}(i, v_i) - \sum_{c_{ij} \in C} (a_{ij}(i, v_i) + a_{ij}(j, v_j)) \\
& = c_\emptyset + \sum_{i=1}^{n} c_i(v_i) + \sum_{c_{ij} \in C} c_{ij}(v_i, v_j).
\end{aligned}
$$

Remarque 3 *La valeur de* $c_\emptyset$ *obtenue après avoir effectuer les opérations op1, op2,..., op5 est une borne inférieure. Cela est assuré par le fait que* $Ax \leq b + \eta$ *est valide : après transformation on a* $c_i(k) \geq 0 \; \forall X_i \in X, \forall k \in D_i$ *et* $c_{ij}(k, l) \geq 0 \; \forall c_{ij} \in C, \forall (k, l) \in D_i \times D_j$.

4.2 Recherche optimale de systèmes d'inégalités valides

Tout système d'inégalités valides peut être utilisé pour transformer un WCSP en un WCSP équivalent. Il peut être intéressant de choisir celui qui permet de calculer la plus grande borne inférieure, c'est-à-dire celui qui permet de maximiser la quantité $\sum_{i=1}^{n} u_i - \sum_{c_{ij} \in C} b_{ij}$ et qui ne produit pas de valuations négatives. Pour cela le programme linéaire suivant peut être résolu :

$$
PL^* \begin{cases}
\max \sum_{i=1}^{n} u_i - \sum_{c_{ij} \in C} b_{ij} \\
a_{ij}(i, k) + a_{ij}(j, l) \leq b_{ij} + c_{ij}(k, l) \; \forall c_{ij} \in C, \\
\qquad\qquad \forall (k, l) \in D_i \times D_j \\
u_i - \sum_{j : c_{ij} \in C} a_{ij}(i, k) \leq c_i(k) \; \forall X_i \in X, \forall k \in D_i
\end{cases}
$$

Les contraintes de ce programme linéaire expriment la condition de ne pas produire de valuations négatives pour que la valeur de $c_\emptyset$ qu'on obtient après les opérations de transformation soit une borne inférieure. Les contraintes de type $a_{ij}(i, k) + a_{ij}(j, l) \leq b_{ij} + c_{ij}(k, l)$ sont celles qui expriment la validité des inégalités du système à utiliser pour la transformation.

Théorème 3 *La valeur* $V(PL^*)$ *du programme* PL^* *est égale à la borne inférieure OSAC.*

Démonstration 3 *Puisqu'on doit toujours satisfaire les contraintes d'affectation* $\sum_{k \in D_i} x_i(k) = 1$ *et* $\sum_{l \in D_j} x_j(l) = 1$, *l'ensemble* $\{ \sum_{k \in D_i} (M + a_{ij}(i, k)) x_i(k) + \sum_{l \in D_j} (M + a_{ij}(j, l)) x_j(l) \leq 2M + b_{ij} + \eta_{ij} : M \in \mathbb{R} \}$ *est constitué d'inégalités toutes équivalentes. Ainsi, si dans une inégalité valide associée à une contrainte* $c_{ij} \in C$ *le coefficient* b_{ij} *n'est pas nul alors on peut choisir* $M = -\frac{bij}{2}$ *pour obtenir une inégalité valide équivalente telle que* $b_{ij} = 0$. *On en déduit que si on fixe de cette manière dans* PL^* *tous les* b_{ij} *à 0 alors la valeur optimale restera la même. Cette fixation transforme exactement* PL^* *au programme linéaire utilisé pour le calcul de OSAC [2] qui est le dual de KPL.*

L'inconvénient de la formulation PL^* réside dans sa taille ($O(ed+n)$ variables et $O(ed^2+nd)$ contraintes) qui fait que sa résolution consomme beaucoup de temps de calcul.

4.3 Recherche approchée de bons systèmes d'inégalités valides

La résolution de PL^* permet de rechercher parmi tous les systèmes contenant une inégalité valide par contrainte du WCSP un qui est optimal : un qui permet de transformer le WCSP en un WCSP équivalent et d'augmenter au maximum la borne inférieure $c_\emptyset$. Cette recherche optimale est coûteuse en temps de calcul. Pour atténuer cet inconvénient, nous proposons d'utiliser un autre système d'inégalités valide qui n'est pas forcément optimal pour transformer le WCSP.

Algorithm 1 Décomposition d'une contrainte

Require: une contrainte c_{ij} du WCSP
Ensure: p_{ij} contraintes binaires $c_{ij}^1, c_{ij}^2, ..., c_{ij}^{p_{ij}}$: $c_{ij}(k, l) = \sum_{q=1}^{p_{ij}} c_{ij}^q(k, l) \; \forall (k, l) \in D_i \times D_j$

1: STOP $\leftarrow$ faux, $p_{ij} \leftarrow 0$
2: **while** STOP = faux **do**
3: **if** $\exists (k, l) \in D_i \times D_j : c_{ij}(k, l) > 0$ **then**
4: $\alpha \leftarrow \min\{c_{ij}(k, l) : (k, l) \in D_i \times D_j, \; c_{ij}(k, l) > 0\}$
5: $p_{ij} \leftarrow p_{ij} + 1$
6: $c_{ij}^p(k, l) \leftarrow \min(c_{ij}(k, l), \alpha), \forall (k, l) \in D_i \times D_j$
7: $c_{ij}(k, l) \leftarrow \max(0, c_{ij}(k, l) - \alpha), \forall (k, l) \in D_i \times D_j$
8: **else**
9: STOP $\leftarrow$ vrai
10: **end if**
11: **end while**

Le système que nous proposons est obtenu par la résolution d'un programme linéaire équivalent au WCSP

et dont toutes les contraintes sont de type clique pondérée. Dans le but de former ces cliques, chaque contrainte $c_{ij} \in C$ du WCSP est d'abord décomposée en plusieures contraintes, chacune d'entre elles associe 0 ou le même coût pour chaque couple de $D_i \times D_j$. Cette décomposition est réalisée par l'algorithme 1. Il produit au plus d^2 contraintes, où d est la taille du plus grand domaine.

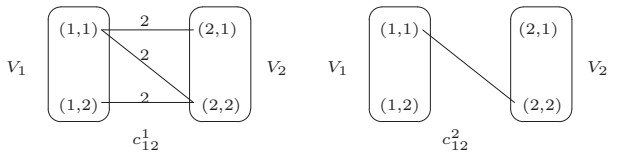

FIG. 3 – Résultat de la décomposition de la contrainte sur la figure 2.

Exemple 3 *Les contraintes représentées par la figure 3 sont le produit de la décomposition de la contrainte représentée par la figure 2.*

Une fois la décomposition de toutes les contraintes est effectuée, un ensemble de cliques Γ_c est construit. Cette construction consiste à associer à chaque contrainte c^q_{ij} produite par la décomposition d'une contrainte initiale $c_{ij} \in C$, $d_i + d_j$ cliques pondérées maximales : une pour chaque valeur de chacune des deux variables X_i et X_j. Si k est une valeur de X_i alors nous lui associons par rapport à c^q_{ij} la clique maximale $CM(\{(i,k)\}, j, q)$ construite comme suit :

1. $E_j = \{(j,l) : l \in D_j, c^q_{ij}(k,l) \neq 0\}$;
2. $E_i = \{(i,k') : k' \in D_i, c^q_{ij}(k',l) \neq 0 \, \forall (j,l) \in E_j\}$;
3. $CM(\{(i,k)\}, j, q) = E_i \cup E_j$.

De même, Si l est une valeur de X_j alors nous lui associons par rapport à c^q_{ij} la clique maximale $CM(\{(j,l)\}, i, q)$ construite comme suit :

1. $E_i = \{(i,k) : k \in D_i, c^q_{ij}(k,l) \neq 0\}$;
2. $E_j = \{(j,l') : l' \in D_j, c^q_{ij}(k,l') \neq 0 \, \forall (i,k) \in E_i\}$;
3. $CM(\{(j,l)\}, i, q) = E_i \cup E_j$.

À chaque contrainte c^q_{ij} nous associons une variable η^q_{ij} et nous formulons linéairement chaque clique Γ_{ij} de Γ_c par une inégalité de type (1) dont b_{ij} est choisi égal à l'unique étiquette que cette contrainte associe aux couples $(k,l) \in D_i \times D_j$:

- $b_{ij} = \max_{(k,l) \in D_i \times D_j} c^q_{ij}(k,l)$;
- $a_{ij}(i,k) = b_{ij}$ si $(i,k) \in \Gamma_{ij}$, $a_{ij}(i,k) = 0$ sinon ;
- $a_{ij}(j,l) = b_{ij}$ si $(j,l) \in \Gamma_{ij}$, $a_{ij}(j,l) = 0$ sinon.

Remarque 4 *Dans le cas des Max-CSP, l'ensemble Γ_c contient au plus $2ed$ cliques. Il en contient au plus $2ed^3$ dans le cas plus général des WCSP : l'algorithme 1 produit au plus d^2 contraintes.*

Exemple 4 *La formulation des cliques pondérées que nous associons aux seules contraintes c^1_{12} et c^2_{12} sur la figure 3 s'écrit :*

$$
\begin{array}{l}
2x_1(1) + 2x_2(1) + 2x_2(2) \leq 2 + \eta^1_{12} \\
2x_1(2) + 2x_2(2) \leq 2 + \eta^1_{12} \\
2x_1(1) + 2x_2(1) \leq 2 + \eta^1_{12} \\
2x_1(1) + 2x_1(2) + 2x_2(2) \leq 2 + \eta^1_{12} \\
x_1(1) + x_2(2) \leq 1 + \eta^2_{12} \\
x_1(2) \leq 1 + \eta^2_{12} \\
x_2(1) \leq 1 + \eta^2_{12} \\
x_1(1) + x_2(2) \leq 1 + \eta^2_{12}
\end{array}
$$

Cette formulation est associée à la seule contrainte c_{12} sur la figure 2. Nous associons une formulation analogue à chaque contrainte initiale (avant décomposition) du WCSP.

Nous notons $IP(\Gamma_c)$ le système linéaire complet qui consiste à chercher une solution qui minimise la quantité $\sum_{i=1}^{n} \sum_{k \in D_i} c_i(k) x_i(k) + \sum_{c_{ij} \in C} \sum_{q=1}^{p_{ij}} \eta^q_{ij}$ et qui satisfait l'ensemble des cliques pondérées ainsi formulées et le système de contraintes S, avec p_{ij} est le nombre de contraintes produites par la décomposition de c_{ij}. Le système $IP(\Gamma_c)$ est du type $IP(A,b)$ dont les inégalités valides sont toutes des cliques pondérées.

Après la construction de $IP(\Gamma_c)$, sa relaxation continue $PL(\Gamma_c)$ est résolue et les valeurs des variables duales obtenues sont récupérées et utilisées pour agréger les inégalités de $PL(\Gamma_c)$ dans le but de produire une seule inégalité valide par contrainte du WCSP.

Exemple 5 *Supposons que c_{12} représentée par la figure 2 est une contrainte parmi les e contraintes d'un WCSP. Soit $IP(\Gamma_c)$ le système linéaire construit après la décomposition de ces e contraintes et soit λ une solution duale optimale de sa relaxation continue $PL(\Gamma_c)$. Le vecteur λ contient une composante pour chaque inégalité dans $IP(\Gamma_c)$. Soit $(\frac{1}{2}, 0, 0, 1, 0, 0, 0, 0)$ les composantes de ce vecteur qui correspondent aux inégalités associées à c_{12}. Ces dernières sont donc agrégées comme suit :*

$$
\begin{array}{lll}
 & \frac{1}{2}. & \left(2x_1(1) + 2x_2(1) + 2x_2(2) \leq 2 + \eta^1_{12}\right) \\
+ & 0. & \left(2x_1(2) + 2x_2(2) \leq 2 + \eta^1_{12}\right) \\
+ & 0. & \left(2x_1(1) + 2x_2(1) \leq 2 + \eta^1_{12}\right) \\
+ & 1. & \left(2x_1(1) + 2x_1(2) + 2x_2(2) \leq 2 + \eta^1_{12}\right) \\
+ & 0. & \left(x_1(1) + x_2(2) \leq 1 + \eta^2_{12}\right) \\
+ & 0. & \left(x_1(2) \leq 1 + \eta^2_{12}\right) \\
+ & 0. & \left(x_2(1) \leq 1 + \eta^2_{12}\right) \\
+ & 0. & \left(x_1(1) + x_2(2) \leq 1 + \eta^2_{12}\right) \\
\hline
= & & 3x_1(1) + 2x_1(2) + x_2(1) + 3x_2(2) \leq 3 + \eta^1_{12} + \eta^2_{12}
\end{array}
$$

En effectuant le changement de variables $\eta_{ij} = \eta^1_{ij} + \eta^2_{ij}$, l'inégalité obtenue peut s'écrire
$3x_1(1) + 2x_1(2) + x_2(1) + 3x_2(2) \leq 3 + \eta_{12}$.

À présent nous disposons d'un système contenant une inégalité valide par contrainte $c_{ij} \in C$ initiale (avant décomposition) du WCSP. Ce système est utilisé pour transformé le WCSP en un WCSP équivalent.

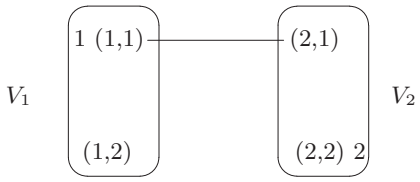

FIG. 4 – Résultat de la transformation de la contrainte sur la figure 2.

Exemple 6 *En utilisant l'inégalité valide* $3x_1(1) + 2x_1(2) + x_2(1) + 3x_2(3) \leq 3 + \eta_{12}$ *et en effectuant les 5 opérations de transformation op1, op2,..., op5 sur la contrainte* c_{12} *de la figure 2 on obtient la contrainte équivalente donnée par la figure 4.*

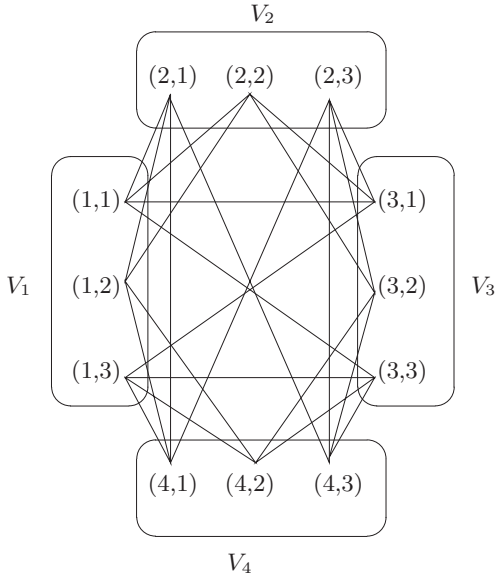

FIG. 5 – Exemple de Max-CSP.

Exemple 7 *La figure 5 montre un exemple de graphe* $G(V, A)$ *représentant un Max-CSP avec 4 variables* $(X = \{X_1, X_2, X_3, X_4\})$ *qui ont le même domaine* $\{1, 2, 3\}$ *et 6 contraintes* $(C = \{c_{12}, c_{13}, c_{14}, c_{23}, c_{24}, c_{34}\})$. *L'ensemble* V *est composé des 4 sous-ensembles* V_1, V_2, V_3 *et* V_4 *associés aux variables : à chaque variable* X_i *est associé l'ensemble* $V_i = \{(i, 1), (i, 2), (i, 3)\}$ *de sommets. L'ensemble* A *est composé des arêtes du graphe : l'existence d'une arête entre deux sommets* (i, k) *et* (j, l) *signifie que l'affectation* $(X_i = k, X_j = l)$ *viole la contrainte* c_{ij}, *i.e,* $c_{ij}(k, l) = 1$.

Observons que ce Max-CSP est complètement arc-consistant (il est FAC);* $\forall X_i \in X, \forall k \in D_i, \forall c_{ij} \in C, \exists l \in D_j : c_j(l) + c_{ij}(k, l) = 0$. *Ainsi, toutes les méthodes qui se basent sur la consistance d'arc calculent la borne inférieure triviale 0. La résolution de la relaxation continue* KPL *de la formulation* $KPLNE$ *de ce Max-CSP permet de calculer une borne inférieure égale à 2.*

L'ensemble Γ_c *correspondant à cet exemple est composé de 36 cliques maximales. Après la résolution de* $PL(\Gamma_c)$ *on peut récupérer une solution duale optimale. L'utilisation de cette solution pour l'agrégation des cliques nous permet d'obtenir le système suivant :*

$$\begin{cases} \min \eta_{12} + \eta_{13} + \eta_{14} + \eta_{23} + \eta_{24} + \eta_{34} \\ t.q \quad x_1(1) + x_1(2) + x_2(1) + x_2(2) \leq 1 + \eta_{12} \\ \qquad x_1(1) + x_1(3) + x_3(1) + x_3(3) \leq 1 + \eta_{13} \\ \qquad x_1(2) + x_1(3) + x_4(1) + x_4(2) \leq 1 + \eta_{14} \\ \qquad x_2(2) + x_2(3) + x_3(1) + x_3(2) \leq 1 + \eta_{23} \\ \qquad x_2(1) + x_2(3) + x_4(1) + x_4(3) \leq 1 + \eta_{24} \\ \qquad x_3(2) + x_3(3) + x_4(2) + x_4(3) \leq 1 + \eta_{34} \\ \qquad x \in S, \eta \geq 0 \end{cases}$$

L'utilisation de ce système pour transformer le Max-CSP initial permet d'aboutir au WCSP représenté par la figure 6. La technique proposée dans [2] permet elle aussi d'aboutir au même résultat puisque elle se base sur la résolution du dual de la formulation KPL. *Cependant, pour les problèmes de grande taille, cette technique nécessite beaucoup plus de temps de calcul.*

Remarque 5 *Toute inégalité valide de type (1) peut être transformée en une autre inégalité valide dont les coefficients sont tous entiers comme suit :*

1. $b_{ij} \leftarrow \lceil b_{ij} \rceil$;
2. $a_{ij}(j, l) \leftarrow \lceil a_{ij}(j, l) \rceil \ \forall l \in D_j$;
3. $a_{ij}(i, k) \leftarrow b_{ij} + \min\{c_{ij}(k, l) - a_{ij}(j, l), l \in D_j\} \ \forall k \in D_i$.

Ainsi, si $Ax \leq b + \eta$ *est un système valide dont les inégalités contiennent des coefficients fractionnaires et si on effectue les transformations ci-dessus à chacune de ces inégalités alors on obtiendra un autre système valide qu'on peut utiliser pour transformer le WCSP en un WCSP équivalent et ne contenant que des valuations entières. La transformation d'un WCSP en un WCSP équivalent et ne contenant que des valuations entières est nécessaire dans le cas où la méthode utilisée pour la résolution globale n'est pas adaptée aux cas des WCSP avec des valuations fractionnaires. L'inconvénient de se restreindre à utiliser que des inégalités à coefficients entiers peut être atténué par la résolution à la place du WCSP initial de celui qu'on obtient*

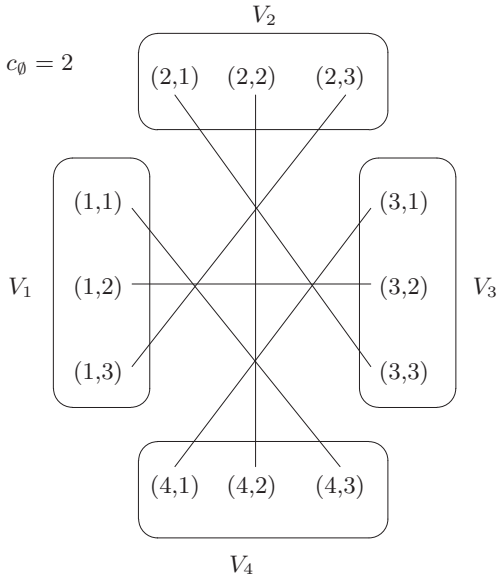

FIG. 6 – Résultat de la transformation du Max-CSP donné par la figure 5.

par la multiplication de toutes les valuations par une même constante entière suffisamment grande.

4.4 Récapitulation

Les points suivants résument les étapes à suivre pour pouvoir bénéficier le plus possible du prétraitement proposé :

1. multiplier par une constante M suffisamment grande les valuations de toutes les contraintes du WCSP (voir la remarque 5) ;
2. décomposer les contraintes du WCSP par l'algorithme 1 ;
3. former un système de cliques Γ_c complet ;
4. résoudre le programme $PL(\Gamma_c)$ et récupérer les valeurs des variables duales ;
5. utiliser les valeurs des variables duales récupérées pour agréger les inégalités valides de $PL(\Gamma_c)$ et former ainsi un système contenant une inégalité valide par contrainte initiale du WCSP ;
6. transformer ces inégalités en inégalités dont tous les coefficients sont des entiers (voir la remarque 5) ;
7. utiliser les inégalités ainsi obtenues pour transformer le WCSP dont les contraintes sont toutes multipliées par M en un WCSP équivalent ;
8. résoudre le WCSP enfin obtenu par une méthode de résolution complète (la plus puissante qui existe de préférence : MEDAC) ;
9. diviser par M le coût de la solution obtenue pour obtenir son coût par rapport au WCSP initial.

Nous noterons par $PL(\Gamma_c) + MEDAC$ la méthode de résolution globale obtenue.

5 Expérimentations

Cette section présente les expérimentations réalisées.

5.1 Bornes inférieures

Nous avons comparé les différentes bornes inférieures à base d'inégalités valides avec des bornes inférieures qui se basent sur l'arc-consistance. Les premiers problèmes testés sont des Max-CSP générés de manière aléatoire[2]. Six classes de Max-CSP ont été considérées. Chaque classe est définie par quatre paramètres $< n, d, e, t >$ tels que n est le nombre de variables ($n = |X|$), d est la taille des domaines ($d = d_i = |D_i|, \forall X_i \in X$), e est le nombre de contraintes ($e = |C|$) et t est la dureté des contraintes ($t = |\{(k,l) : (k,l) \in D_i \times D_j, c_{ij}(k,l) = 1\}|, \forall c_{ij} \in C$).

Les graphes de contraintes des six classes sont de trois types :

1. peu denses (`sparse (S)`) : $e = 2.5n$;
2. denses `(D)` : $e = \frac{n(n-1)}{8}$;
3. complets `(C)` : $e = \frac{n(n-1)}{2}$.

Pour des valeurs de n, d et e fixées, il existe une dureté minimale t^0 des contraintes à partir de laquelle les problèmes générés sont sur-contraints et difficiles à résoudre. Les types de duretés considérées sont :

1. faible (`loose (L)`) : $t = t^0$;
2. forte (`tight (T)`) : $t = d^2 - 0.25t^0$.

En combinant les trois types de densité des graphes de contraintes avec les deux types de dureté des contraintes on obtient six classes : SL, ST, DL, DT, CL et CT. Dans chaque classe, le paramètre d (taille des domaines) est fixé à 10. Le nombre de variables n est fixé à 40 pour les problèmes de la classe SL, à 30 pour les problèmes de la classe DL, à 25 pour les problèmes des classe ST, DT et CL et à 15 pour les problèmes de la classe CT. Pour chaque classe, 50 instances de problèmes sont considérées[3].

Les bornes inférieures à base d'inégalités valides que nous avons évaluées sont :

1. $V(PL(\Gamma_c))$: borne inférieure obtenue par la résolution de $PL(\Gamma_c)$;

[2] Le protocole de génération est défini dans [8].

[3] Ces Max-CSP sont disponibles à http ://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/softCSP.

2. $V(PL^*)$: borne inférieure obtenue par la résolution de PL^*. C'est la même borne obtenue par la résolution de KPL ou par le calcul de la borne $OSAC$.

Ces bornes sont comparées avec la borne EDAC*.

L'outil utilisé pour la résolution des deux systèmes linéaires est le solveur *Ilog Cplex*. Nous avons utilisé la méthode barrier qui est plus rapide en moyenne sur ces problèmes. Le logiciel libre *toolbar* [3] est utilisé pour le calcul de la borne EDAC*.

Pour les problèmes dont la dureté des contraintes est faible (classes SL, DL et CL), toutes les méthodes calculent la borne inférieure triviale 0.

Pour les classes ST, DT et CT, le tableau suivant reporte la borne inférieure moyenne calculée par chacune des techniques citées.

	$EDAC^*$	$V(PL(\Gamma_c))$	$V(PL^*)$
ST	4.26	11	12.30
DT	9.96	18.30	19.80
CT	44.46	53.89	54.75

Le tableau suivant reporte les temps moyens en secondes que nécessitent les résolutions des programmes $PL(\Gamma_c)$ et PL^*, les temps que nécessitent le calcul de EDAC* est de l'ordre de 10^{-6} secondes[4].

	SL	DL	CL	ST	DT	CT
$PL(\Gamma_c)$	0.29	0.1	0.15	0.09	0.10	0.12
PL^*	1.97	0.84	11.80	0.34	0.55	1.63

On remarque que pour les problèmes dont la dureté des contraintes est forte, les bornes calculées par la résolution de $PL(\Gamma_c)$ et PL^* sont largement supérieures à $EDAC*$.

Les bornes $V(PL(\Gamma_c))$ et $V(PL^*)$ sont proches. Le calcul de $V(PL(\Gamma_c))$ présente l'avantage d'être moins gourmand en temps de calcul.

Les bornes $EDAC^*$, $V(PL(\Gamma_c))$ et $V(PL^*)$ sont aussi évaluées sur des WCSP réels. Les WCSP testés sont des instances du problème d'affectation de fréquences aux liens radio (*Radio Link Frequency Assignment Problem of the CELAR* [1]). Les instances considérées sont scen07reduc.wcsp, scen08reduc.wcsp, graph11reducmore.wcsp et graph13reducmore.wcsp (voir la section Benchmarks de [3]). Le tableau suivant reporte pour chacun de ces problèmes les meilleures bornes inférieures[5] et supérieures[6] connues à ce jour.

	scen07	scen08	graph11	graph13
Bsup	343592	262	3080	10110
Binf	300000	216	3016	9925

[4]Nous avons utilisé une machine cadencée à 2,80 GHz. Elle est équipée d'une RAM de 2 Go.

[5]Elles sont calculées par des méthodes de complexité temporelle exponentielle.

[6]Elles sont calculées par des méthodes de recherche locale.

Le tableau suivant donne pour chaque problème et chaque méthode de calcul la borne inférieure calculée et le temps de calcul en secondes.

	scen07	
	Binf	Temps (s)
$V(PL(\Gamma_c))$	30346.7	118.66
$V(PL^*)$	31453.1	1461.94
EDAC*	10000	0.01
	scen08	
	Binf	Temps (s)
$V(PL(\Gamma_c))$	44.1278	167.88
$V(PL^*)$	48.3225	1999.21
EDAC*	6	0.04
	graph11	
	Binf	Temps (s)
$V(PL(\Gamma_c))$	2952.34	50.94
$V(PL^*)$	2957	109.87
EDAC*	2710	0.01
	graph13	
	Binf	Temps (s)
$V(PL(\Gamma_c))$	9797.5	147.01
$V(PL^*)$	9797.5	2031.51
EDAC*	8722	0.03

Ces résultats montrent que la résolution des modèles $PL(\Gamma_c)$ et PL^* permettent de calculer des bornes inférieures proches ou même égales dans le cas de l'instance graph13. Le temps de résolution de PL^* est beaucoup plus important que celui de $PL(\Gamma_c)$. Ces temps sont obtenus en utilisant la méthode barrier pour la résolution de programmes linéaires. Les problèmes scen07 et graph11 sont de taille relativement moins importante par rapport aux tailles des deux autres problèmes. La méthode duale du simplexe résout plus rapidement les programmes $PL(\Gamma_c)$ qui correspondent aux problèmes scen07 et graph11. Elle met 57.64 secondes pour résoudre scen07 et 16.32 secondes pour résoudre graph11.

Par rapport à $EDAC^*$, $V(PL(\Gamma_c))$ et $V(PL^*)$ sont largement supérieures. Pour les instances graph11 et graph13, les bornes $V(PL(\Gamma_c))$ et $V(PL^*)$ sont même proches des meilleures bornes inférieures qui sont connues à ce jour et qui sont obtenues par des méthodes de calcul exponentielles.

5.2 Méthodes de résolution

Ce paragraphe présente les résultats de l'évaluation de la méthode de résolution $PL(\Gamma_c) + MEDAC$.

Les problèmes testés sont les Max-CSP des six classes décrites au paragraphe précédent. Le tableau suivant résume les résultats obtenus.

	MEDAC		$PL(\Gamma_c) + MEDAC$	
	#nœuds	Temps (s)	#nœuds	Temps (s)
SL	24872.1	2.605	24872.1	2.762
DL	14706	1.376	14706	1.467
CL	202132	25.515	202132	25.741
ST	7993.38	0.774	2440.88	0.601
DT	22025	2.374	5856.34	1.601
CT	156588	16.751	67036.5	16.730

Le prétraitement effectué par $PL(\Gamma_c)+MEDAC$ se base sur la résolution d'un système linéaire $PL(\Gamma_c)$ associé à un ensemble Γ_c de cliques. Les problèmes testés étant des Max-CSP, la décomposition des contraintes n'a pas été nécessaire. La constante M par laquelle nous avons pondéré[7] les inégalités de type clique binaire de $PL(\Gamma_c)$ est choisie égale à 1000. L'utilisation d'une solution duale optimale de $PL(\Gamma_c)$ pour l'agrégation de ses cliques permet de construire un système contenant une inégalité valide par contrainte du Max-CSP. La technique présentée pour transformer un système d'inégalités valides en un autre système d'inégalités valides et dont tous les coefficients sont des entiers dégrade peu la qualité de la borne inférieure[8]. En utilisant le système d'inégalités ainsi obtenu, le problème résultant après transformation est un WCSP dont toutes les valuations sont entières et peut être résolu par $MEDAC$[9].

Sur les problèmes dont la dureté des contraintes est faible, le prétraitement n'apporte aucune contribution à la résolution puisque pour ces problèmes la valeur $V(PL(\Gamma_c))$ est nulle. Il est intéressant quand même de remarquer que le temps perdu pendant ce prétraitement n'est pas significatif par rapport au temps de résolution global. C'est sur les problèmes dont la dureté des contraintes est forte que ce prétraitement est intéressant, surtout sur les problèmes de la classe DT où $PL(\Gamma_c) + MEDAC$ développe 3,76 fois moins de nœuds que $MEDAC$ et met 1,48 fois moins de temps.

6 Conclusion

Dans cet article, nous avons introduit la notion d'inégalité binaire valide et montré qu'elle peut être bien exploitée dans le cadre des Max-CSP et WCSP.

Nous avons proposé une technique à base d'inégalités valides pour transformer un WCSP en un WCSP équivalent. Nous avons par la suite démontré qu'il est toujours possible de construire un système linéaire contenant une seule inégalité valide par contrainte du WCSP et dont la valeur est égale à la borne $OSAC$. La recherche d'un tel système est coûteuse en temps de calcul, nous avons alors proposé d'utiliser un autre système qui n'est pas forcément optimal mais qui est plus facile à construire.

Les résultats numériques préliminaires obtenus sont intéressants. En plus de cet intérêt pratique, l'étude présentée ouvre des voies de recherche intéressantes. En effet, l'utilisation d'inégalités binaires valides peut être généralisée à celle des inégalités valides n-aires.

Nous comptons étendre cette étude au cas des inégalités valides n-aires et au cas des WCSP n-aires.

[7] Cette pondération n'est pas nécessaire si la méthode de résolution complète utilisée est adaptée aux contraintes valuées par des réels.

[8] Plus la valeur de M est grande plus cette dégradation est négligeable.

[9] Comme dans [2], nous avons adopté cette méthode au fait que les coûts des solutions sont des multiples de $M = 1000$: si une solution de coût $3M$ par exemple est trouvée alors elle est sauvegardée et la borne supérieure est fixée à $2M + 1$ pour éliminer toutes les solutions dont le coût dépasse $2M$.

Références

[1] B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J. P. Warners. Radio link frequency assignment. *Constraints*, 4(1) :79–89, 1999.

[2] M. Cooper, S. de Givry, and T. Schiex. Optimal soft arc consistency. In *IJCAI*, 2007.

[3] S. de Givry, F. Heras, J. Larrosa, E. Rollon, and T. Schiex. The softcsp and max-sat benchmarks and algorithm web site. http ://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/softcsp. Technical report, 2006.

[4] S. de Givry, F. Heras, M. Zytnicki, and J. Larrosa. Existential arc consistency : Getting closer to full arc consistency in weighted csps. In *IJCAI*, pages 84–89, 2005.

[5] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4) :373–395, 1984.

[6] M. I. Khemmoudj and H. Bennaceur. Inference de cliques pour la résolution de max-csp. In *2th Journées Francophones de Programmation par Contraintes (Nimes, 2006)*, pages 229–238, 2006.

[7] Arie Koster. *Frequency Assignment : Models and Algorithms*. PhD thesis, Maastricht, Belgium, 1999.

[8] J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted csp. In *IJCAI*, pages 239–244, 2003.

[9] T. Schiex. Arc consistency for soft constraints. In *CP*, volume 1894 of *Lecture Notes in Computer Science*, pages 411–424. Springer, 2000.